

■ Model-Based Design  
for DO-178B (Excerpts)

## Applying Model-Based Design to Safety-Critical Systems

Model-Based Design with Simulink® products enables the use of executable specifications, automated verification and validation, and production code generation—an approach that reduces software development time, improves quality, and enhances innovation.

The benefits of Model-Based Design apply to a variety of embedded systems, including safety-critical applications: Code generated using MathWorks tools and Model-Based Design has been certified on safety-critical systems, and developers report six-sigma software quality and 50% improvements in productivity [1,2].

To apply Model-Based Design successfully to safety-critical system development requires extra consideration and rigor because these systems must adhere to tightly defined standards.

DO-178B, “Software Considerations in Airborne Systems and Equipment Certification [3]” is one of the most widely used of these standards. It was published in 1992, when most software was hand-coded. As a result, it does not cover advanced software development technologies, and must be mapped onto the processes and tools used in Model-Based Design.

### About this Guide

A 100+-page paper, *Model-Based Design for DO-178B*, explains in detail how to meet the requirements in the DO-178B standard using Model-Based Design with Simulink products (see sidebar).

This Guide includes three excerpts from *Model-Based Design for DO-178B*:

- Introduction
- DO-178B Software Lifecycle
- Model Architecture Considerations

### Model-Based Design for DO-178B

This 100+-page paper covers the following topics:

- DO-178B software lifecycle
- Model architecture
- Solvers
- Data import/export
- Optimization
- Model diagnostics
- Hardware implementation
- Code generators
- Block selection
- Block settings and data types
- Stateflow® software
- Run-time library considerations

To obtain a copy, visit  
[www.mathworks.com/DO-178B](http://www.mathworks.com/DO-178B)

Note your Simulink® and Real-Time Workshop® Embedded Coder version information (using the MATLAB `ver` command) and indicate that you are willing to provide feedback on this document for your DO-178B project.

## EXCERPT 1

### 1. Introduction

*The purpose of this document is to provide approaches for applying MathWorks products for Model-Based Design to safety-critical or mission-critical systems that must meet DO-178B objectives for certification. Approaches presented in this document are provided as recommendations, but are not the only methods that could be used in meeting DO-178B objectives for certification. These guidelines may also be applied to safety-critical or mission-critical systems that do not have to meet the objectives of DO-178B, but this document does assume that the software lifecycle follows that standard.*

*Disclaimer: While adhering to the recommendations in this document will reduce the risk that an error is introduced in the software development process and is not detected, it is not a guarantee that the system being developed will be safe. Conversely, if the recommendations in this document are not followed, it does not mean that the system being developed will be unsafe.*

*DO-178B defines five software levels: A, B, C, D and E. Systems being developed to levels A or B would certainly fall into the category of safety-critical or mission-critical, because failures of these systems could result in loss of life. Systems being developed to level C may result in increased crew workload and a reduction in safety and should therefore provide a high degree of reliability. The recommendations in this document are applicable to systems being developed to levels A, B and C.*

*This document does not discuss custom S-functions, device drivers, or Embedded MATLAB™ functions that are produced by the developer using manual coding techniques. Hand-written code that is called by generated code will have to meet all of the lifecycle requirements of DO-178B. Considerations for Embedded MATLAB functions may be added to this document in the future.*

## EXCERPT 2

### **2. DO-178B Software Lifecycle**

The following processes make up the DO-178B software lifecycle:

- Planning
- Software development
- Verification of requirements
- Verification of design
- Verification of coding and integration
- Testing of outputs of integration
- Verification of verification process results
- Software configuration management
- Software quality assurance
- Certification liaison

There are objectives that must be met for each of the lifecycle stages defined in DO-178B. These objectives are summarized in Appendix A of DO-178B in the form of tables. The following sections of this document summarize those tables and provide recommendations on how the objectives may be met using Model-Based Design. The potential usage of available tools for Model-Based Design in achieving the objectives is also included.

## 2.2 Table A-2 Software Development Process

The following table contains a summary of the software development process objectives from DO-178B, including description, applicable DO-178B reference sections and software levels the objective is applicable to. The table also provides the tools for Model-Based Design that may be used in satisfying the objectives.

	Description	Section	Software Levels	Available Products for Model-Based Design
1	High-level requirements are developed.	5.1.1a	A, B, C, D	Simulink® Stateflow®
2	Derived high-level requirements are developed.	5.1.1b	A, B, C, D	Simulink® Stateflow®
3	Software architecture is developed.	5.2.1a	A, B, C, D	Simulink® Stateflow®
4	Low-level requirements are developed.	5.2.1a	A, B, C, D	Simulink® Stateflow®
5	Derived low-level requirements are developed.	5.2.1b	A, B, C, D	Simulink® Stateflow®
6	Source code is developed.	5.3.1a	A, B, C, D	Real-Time Workshop® Embedded Coder
7	Executable Object Code is produced and integrated in the target computer.	5.3.1a	A, B, C, D	Not applicable

The following sections describe in more detail the potential impacts as compared to traditional development, if applicable, for each software development process objective when using Model-Based Design.

### 2.2.1 High-level requirements are developed

If models are defined as high-level software requirements, then Simulink and Stateflow software may be used to develop the high level software requirements. The components within these models (such as Simulink blocks or Stateflow objects) would then trace to the appropriate system-level requirements, which are developed in accordance with ARP4754. The models should be developed in accordance with the modeling standards defined during the planning process.

### 2.2.2 Derived high-level requirements are developed

If models are defined as high-level software requirements, then any Simulink or Stateflow components that do not trace to the system requirements would be identified as derived requirements, and these would be provided to the safety assessment process.

### **2.2.3 Software architecture is developed**

Architecture of individual software modules may be defined by the Simulink and Stateflow models, including sequencing and interfacing of the various elements within the models. If model referencing capability is used, then the Model Dependency viewer may be used to document the architecture of the software modules that are integrated using this capability.

The higher level architecture of how the generated code interfaces to other code within the system must be defined separately. This may include interface to the real-time operating system (RTOS), calling sequence for the code generated from the model and data interface to other code modules.

### **2.2.4 Low-level requirements are developed**

If models are defined as low-level software requirements, then Simulink and Stateflow may be used to develop the low-level software requirements. The components within these models would then trace to the appropriate high-level software requirements. The models should be developed in accordance with the modeling standards defined during the planning process.

If the models are defined as high-level software requirements, and source code will be generated directly from those models, then this objective does not apply.

### **2.2.5 Derived low-level requirements are developed**

If models are defined as low-level software requirements, then any Simulink or Stateflow components that do not trace to the high-level software requirements would be identified as derived requirements and these would be provided to the safety assessment process.

If the models are defined as high-level software requirements, then library components or reusable model referencing functions may be considered to be derived low-level requirements.

### **2.2.6 Source code is developed**

Real-Time Workshop® Embedded Coder may be used to generate the source code from the model. The source code can trace to the model components through the use of appropriate commenting options described later in this document. The source code can be generated in accordance with MISRA-C standards, with some exceptions, using appropriate modeling standards.

### **2.2.7 Executable object code is produced and integrated in the target computer**

The generated source code may be compiled and linked using standard compilers/linkers. Real-Time Workshop® Embedded Coder may generate a make file for use by the compiler or this may be developed manually. The executable object code is then loaded onto the target computer.

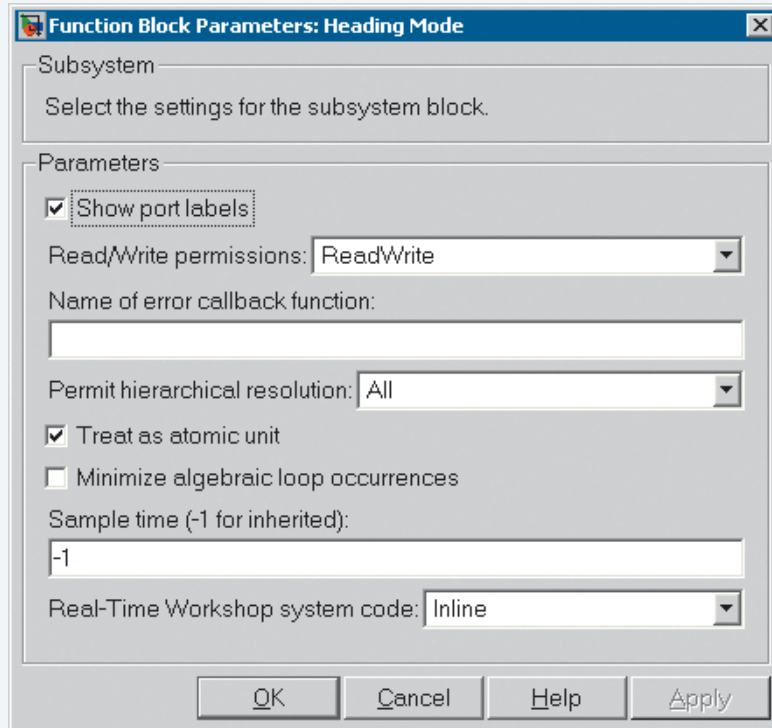
## EXCERPT 3

### 3. Model Architecture Considerations

This section contains recommendations with respect to model architecture considerations when developing safety critical or mission critical systems. The architectural considerations in this section are intended to ease the verification activities associated with the code generated from the models.

#### 3.1 Use of Atomic Subsystems

When creating models that use subsystems to break the models into separate viewable pages, it is recommended that the subsystems be set to “Treat as atomic unit.” See the subsystem parameters dialogue below.



When “Treat as atomic unit” is selected, the generated code for the subsystem is grouped together and typically includes a starting and ending comment in the code file. Having a block of code map directly to a page on the diagram simplifies the code review process. If this option is not selected, then the generated code may be interleaved such that lines of code trace to blocks across multiple pages on the diagram.

Selecting “Treat as atomic unit” also allows Simulink to check and report (as an error or warning) if this group of blocks cannot be executed as a unit, e.g. function call across subsystem boundary, or direct feedback loop, etc.

“Real-Time Workshop system code” should be set as “Inline” or “Function.” Use of “Auto” or “Reusable function” can lead to confusion with respect to requirements traceability in the generated code. If reusable functions are desired, then Model Referencing should be used to instantiate the reusable functions.

### 3.2 Use of Model Referencing

The model referencing feature allows a large model to be broken into smaller separately configured units. This results in separate code files for each of the separate models. There are many advantages to having these separately configured files:

- Each model and its associated code can be verified as a separate component.
- When a reference model changes, only its code has to be reviewed.
- When a reference model changes, only its verification testing has to be performed.
- Regression analysis is greatly simplified.
- Structural coverage analysis is simplified when using smaller models and code.
- Reference model code can be reused.

When using Model Referencing, especially with large teams, configuration management of the reference models is very important. There are two “best practice” scenarios for configuration management of the reference models that are described here.

### References

[1] “Achieving Six Sigma Software Quality Using Automatic Code Generation.”  
[www.mathworks.com/industries/aerospace/miadc05/presentations/potter.pdf](http://www.mathworks.com/industries/aerospace/miadc05/presentations/potter.pdf)

[2] “Alstom Generates Production Code for Safety-Critical Power Converter Control Systems.”  
[www.mathworks.com/products/rtwembedded/userstories.html?file=10591](http://www.mathworks.com/products/rtwembedded/userstories.html?file=10591)

[3] “Software Considerations in Airborne Systems and Equipment Certification.” Document No. RTCA/DO-178B, December 1, 1992, prepared by SC-167.

### Resources

#### VISIT

[www.mathworks.com](http://www.mathworks.com)

#### TECHNICAL SUPPORT

[www.mathworks.com/support](http://www.mathworks.com/support)

#### ONLINE USER COMMUNITY

[www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)

#### DEMOS

[www.mathworks.com/demos](http://www.mathworks.com/demos)

#### TRAINING SERVICES

[www.mathworks.com/training](http://www.mathworks.com/training)

#### THIRD-PARTY PRODUCTS AND SERVICES

[www.mathworks.com/connections](http://www.mathworks.com/connections)

#### WORLDWIDE CONTACTS

[www.mathworks.com/contact](http://www.mathworks.com/contact)

#### E-MAIL

[info@mathworks.com](mailto:info@mathworks.com)